OBJEKTOVÉ METODOLOGIE – JEJICH UŽITÍ A VÝKLAD

Martin Molhanec

České vysoké učení technické – FEL, Technická 2, 166 27 PRAHA 6, Dejvice, ČR
tel.: (++420) 2 2435 2118 mailto: molhanec@fel.cvut.cz, http://martin.feld.cvut.cz/~mmm

Abstrakt
Obsahem příspěvku je kritický pohled autora na výklad objektového paradigmatu, tak jak se s ním měl možnost setkat v pracích studentů i v odborných publikacích. Speciálně bude kritice podroben chybný výklad UML.

1. Úvod

Tento příspěvek navazuje na můj předchozí příspěvek na této konferenci v loňském roce s názvem *UML – několik kritických poznámek* 3. Zdrojem obou příspěvků je moje nespokojenost s výkladem objektového paradigmatu a to zejména při výkladu metody UML. Nebezpečí vidím zejména u studentů, kteří nesprávným pochopením získají nesprávné návyky a bude pak dlouho trvat nežli se jich zbaví. Netvrdím, že má tvrzení jsou vždy správná, ale i tak doufám, že možná vyvolají určitou reakci a diskuzi v této oblasti.

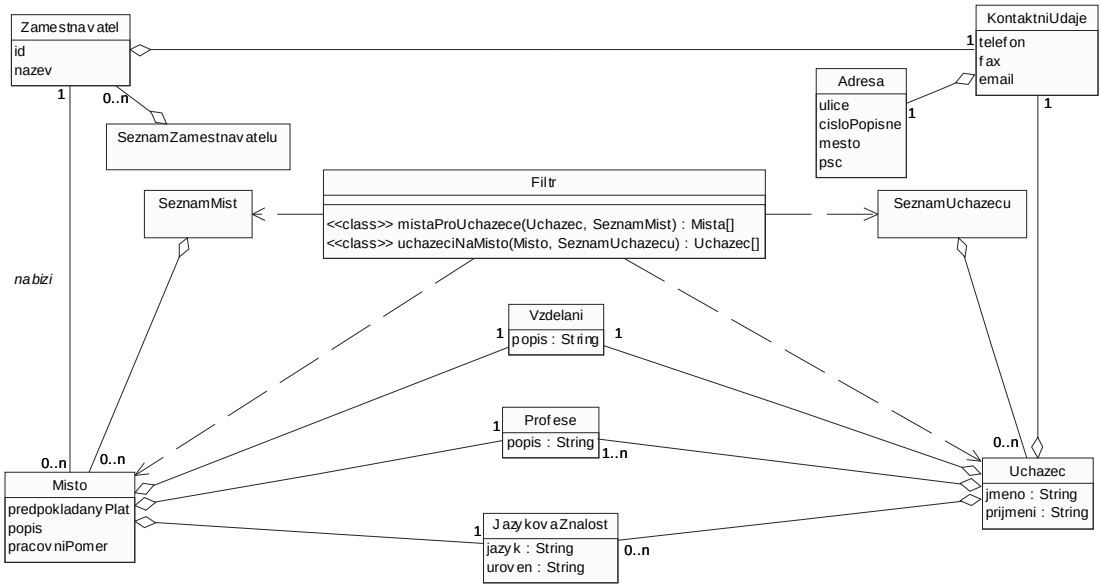
2. Několik špatných příkladů

Pozastavme se nejprve u několika špatných příkladů, respektive u příkladů, které já považuji za špatné a pokusím se vysvětlit proč je považuji za špatné. Podívejme se nejprve na obrázek *obr. 1* na stránce 112. Jedná se o práci studenta prvního ročníku VŠ, který měl za úkol udělat jednoduchou analýzu jednoduché aplikace. Existuje několik dobrých důvodů, proč tento diagram považuji za chybný. Jedná se o aplikaci, která shromažďuje jednak nabídky pracovních míst od zaměstnavatelů a jednak kontaktní údaje na potenciální zájemce o pracovní místa. Všimněme si následujících skutečností.

- Prakticky všechny vztahy až na jediný jsou agregace. To je samozřejmě podezřelé. Problém totiž spočívá ve skutečnosti, že rozdíl mezi agregací a vztahem 1 : M je málokdy správně a dostatečně srozumitelně vysvětlen! Tímto problémem se budu zabývat v podkapitole *Co je to vlastně ten kontejner?*.
- Nepochopení, jak se agregace vlastně kreslí a jakou obsahuje kardinalitu a parcialitu. Ve více nežli jednom případě je uvedena agregace, ve které celek (*Místo*) obsahuje právě jednu část (*Vzdělání*). Pochopitelně, pro zkušeného praktika nezasaženého současným vzděláním, se zde jedná o tzv. číselník se vztahem 1 : M.
- Všimněme si také z analytického hlediska nesmyslných tříd *Seznam {Zaměstnavatelů, Míst, Uchazečů}*! Typický programátor totiž nedokáže pochopit, že jeho implementační pomůcka bez které si nedokáže představit, jak to vlastně v jazyku C++ naprogramuje, je z pohledu analytika naprosto nezajímavá záležitost. Navíc, asi netuší, že nic takového potřebovat nebude, pokud data aplikace budou obsažena v databázi!
- Třída *Filtr* je pochopitelně naprosto nesmyslná v analytickém kontextu, kdy analyzujeme třídy (objekty) reálného světa se kterými aplikace pracuje a nikoliv programátorské konstrukce!

- Pochopitelně, ani vztah *závislosti* (například mezi třídou *Filtr* a *Uchazeč*) nemá dle mého názoru v analytickém diagramu své místo!

obr. 1



Další špatné příklady bohužel nemohu ukázat, protože nemám k dispozici dané texty v elektronické formě. Nicméně se jedná o typické nepochopení skutečnosti, že existuje rozdíl mezi třídami, které představují data se kterými aplikace pracuje a mezi třídami, které představují funkcionalitu a data vlastní aplikace. Je typické, že se jednalo o diplomanty vychované na programovacím jazyku *Smalltalk*, kde zřejmě díky vlastnostem tohoto jazyka při práci s databází ztrácejí studenti schopnost rozpoznat, kdy se jedná o třídu vlastní aplikace a kdy o třídu, která v aplikaci obaluje přístup do tabulek relační databáze. Toto ovšem neznamená, že by jazyk Smalltalk byl špatným jazykem! Ale pouze upozorňuji na skutečnost, že si studenti z oblasti programování přenášejí do oblasti analýzy chybné návyky!

3. Diagram tříd a jeho slabiny

Moje kritika se z celého objektového paradigmatu zaměří speciálně na diagram tříd a jeho vlastnosti. Položme si však otázku, co má vlastně diagram tříd z analytického hlediska představovat? Většina textů o tom, co je to diagram tříd, totiž o tomto nehovoří! Pokud se podíváme na klasické texty o objektových metodologiích vidíme, že se většinou předpokládá, že se diagram tříd použije místo klasického ER (Entity-Relationship) diagramu. Co znázorňuje ER diagram je jasné, jedná se o diagram zachycující vztahy mezi daty se kterými aplikace pracuje! Pokud se tedy díváme na diagram tříd pohledem datového analytika je situace také zcela jasná, diagram tříd zobrazuje datové objekty se kterými aplikace pracuje.

Bohužel existence objektově orientovaných jazyků a UML navádějí k tomu, abychom si pomocí diagramu tříd zobrazovali vztahy mezi třídami aplikace! Ale to je už jiný diagram! To je, jak si můžeme v souladu s moderní češtinou říci, o něčem jiném! A pokud k tomuto faktu přidáme skutečnost, že studenti zmatení *Smalltalkem* a jinými jazyky, kde datové objekty jsou obaleny objektem aplikačním, není potom divu, že toto vše vede k situacím, které jsem zde již uvedl. Vina není ovšem na straně programovacího jazyka! To jenom studenti stále nechápou o čem vlastně ta analýza je! Když k tomu dále připočteme skutečnost, že studenti dříve programují nežli analyzují a sami se něco naučí z různých učebnic UML a na VŠ je často učí

na cvičeních místo odborných asistentů mladí doktorandi, kteří znalostmi nejsou od studentů příliš vzdáleni, nelze se divit, že nejsou sto pochopit, že v analýze nás zajímá objektový model dat se kterými aplikace pracuje, protože a to je nesmírně důležité pochopit, tento model se **nemění,** zatímco třídy a objekty vlastní aplikace se mění podle toho, kdo jí programuje a jaký programovací jazyk použijeme!

4. Vztahy a to nikoliv mezi lidmi

Další skutečnost, která mne irituje jsou šipky! V diagramu tříd, který představuje vztahy mezi daty se kterými aplikace pracuje, nejsou žádné šipky potřeba! Pochopitelně je tento zlozvyk způsoben UML, které kreslení šipek v diagramu tříd dovoluje, ale je nutné si uvědomit ve kterých souvislostech!

- **Navigace** je první případ, kdy je možné v diagramu tříd UML nakreslit šipku. Jedná se o šipku, která se kreslí na normální spojnici, která slouží pro označení vztahu. Jaký je význam *navigace*? Pokud se podíváme do reference UML je zcela jasné, že se jedná o konstrukt, který má význam pouze implementační a nikoliv analytický! Navigace se totiž vysvětluje tak, že ve směru šipky je možný přístup od jednoho objektu ke druhému. V referenci se uvádí, že navigace je realizována pomocí *pointerů*. Je zřejmé, že navigace může být užitečná až ve fázi, kdy si potřebujeme nakreslit implementační diagram v tom případě, kdy jsou data aplikace uložena například v paměti počítače anebo v nějakém druhu databáze s navigačním přístupem. Pokud se pohybujeme v oblasti analýzy či je implementace realizována pomocí databáze s přístupem k datům prostřednictvím dotazovacího jazyka, je zřejmé, že navigace je naprosto zbytečným konstruktem!

- **Závislost** je druhý standardní případ, kdy je možné v diagramu tříd UML a nejen v tomto, kreslit šipku. Tentokrát se jedná o zvláštní čárkovanou šipku (tedy nikoliv šipku na některém standardním vztahu), která směřuje od třídy závislé ke třídě na které závislá třída závisí. V referenci se závislost vysvětluje tak, že změna v nezávislé třídě může vyvolat změnu ve třídě závislé. Bohužel i v samotné referenci je závislost vysvětlena poněkud nejednoznačně. Každopádně se uvádí několik významů závislosti, například *abstraction*, *binding*, *permission*, *usage*, a další. Pro jednodušší vysvětlení závislosti se také uvádí, že objekt nezávislé třídy je používán jako argument metody třídy závislé. Domnívám se, že v oblasti datové analýzy se bez závislostí obejdeme.

Pokud se jedná o *zprávy* (*messages*) jejichž grafické zobrazení obsahuje také šipku, tak ty do diagramu tříd, který zobrazuje *statické* vztahy mezi třídami, pochopitelně nepatří.

5. Co je to vlastně ten kontejner?

Další problém v pochopení objektových metod je nejasné rozlišení mezi vztahem 1 : M a vztahem, který se v OOA nazývá *celek-část* a v UML *agregace* či *kompozice*. Také hovoříme o vztahu typu kontejner nebo jednoduše o vztahu *HAS A*. Dle UML reference (1,2) je kompozice silnější typ agregace. Rozdíl mezi agregací a kompozicí je ten, že v případě kompozice nemůže *část* existovat samostatně! Problém však stále spočívá v odlišení mezi vztahem 1 : M a agregací či kompozicí. Samotná reference moc dobrý návod neposkytuje, mluví totiž o tom, že rozlišení závisí na *matter of taste*, což nám skutečně moc nepomůže ☺ Nicméně jeden dost dobrý návod jsem nalezl v jiné literatuře. Celek a jeho část musí mít určitou fyzickou nebo obdobnou souvislost. Pokud si nejsme jisti zkusme uvažovat o tom, že

část přemístíme, pokud se přitom přemísťuje i *celek* jedná se o agregaci či kompozici, pokud nikoliv, jde o vztah 1 : M.

6. Základní otázka objektů, tříd, vztahů a tak vůbec …

Přestože pojem objektového paradigmatu existuje již celou řadu let a první velké práce v oblasti objektově orientované analýzy vznikly již na počátku let 90, není podobně jako u jiných velkých *témat* dosaženo sjednocení všech názorů. Ač velice nerad musím například ne zcela úplně souhlasit s výkladem souvislostí mezi objekty, vztahy a zprávami ve velice inspirativní publikaci mého kolegy a přítele V. Merunky 4. (Ostatně, jedná se o jednu z mála publikací, které se problematice analýzy kvalifikovaně věnují). Na straně druhé se jedná spíše o rozdíl ve *filozofickém* přístupu k celé problematice. Například – byl dříve vztah či zpráva, existuje skládání v analytické úrovni? Ve skutečnosti, však s ohledem na naše zkušenosti, přes výše zmíněné rozdíly, bude výsledek naší analýzy nějaké konkrétní skutečnosti pravděpodobně téměř shodný. Otázka je, do jaké míry jsou tyto jemné rozdíly v analytickém přístupu správně pochopit a interpretovat naši studenti.

Je snad typickým paradoxem, že čím více člověk o určité problematice ví, tím více pochybuje, že ji rozumí. A proto lze s úspěchem pochybovat, že definitivní řešení správného výkladu a pochopení objektově orientované paradigmatu bude definitivní.

7. Shrnutí

Cílem tohoto krátkého příspěvku bylo naznačit několik problémů, které se vyskytují při používání a výkladu objektově orientovaných diagramů tříd v oblasti analýzy informačních systémů. Možná můj postoj vyplývá z mé předchozí databázové zkušenosti, ale každopádně jsem přesvědčen o tom, že datová analýza, přestože prováděná objektově orientovaným způsobem, je stále platným a nutným nástrojem pro pochopení a návrh informačních systémů. Nejdůležitější teze, které jsem chtěl v tomto článku vyjádřit jsou následující.

- Diagram tříd se v oblasti analýzy zabývá daty se kterými aplikace pracuje a nikoliv objekty vlastní aplikace.
- Diagram tříd neobsahuje žádné šipky, protože je v něm nepotřebujeme! Ale vždy potřebujeme znát kardinalitu a parcialitu!
- Je dobré dobře vědět jaký je rozdíl mezi agregací či kompozicí a vztahem 1 : M.
- Předchozí znalost UML diagramů pro oblast implementace je pro oblast analýzy spíše nežádoucí.
- Programátorská zkušenost z jazyků podobných *Smalltalku* je také spíše na závadu.
- Programování může učit skoro každý, ale učit analýzu chce zkušenosti!
- V učebnicích UML se málokdy rozlišuje mezi užitím UML ve fázi analýzy a ve fázi implementace.
- Je nutné pečlivě číst referenční manuál od vlastních autorů UML.

Doufám, že tento článek vyvolá širší diskusi o objektově orientované analýze, jejího použití a výkladu. Možná, že i mé názory nejsou vždy zcela správné. Každopádně si objektově orientovaná analýzy zaslouží, aby byla správně chápána a vykládána.

Literatura:

1. Booch, G., Jacobson, I., Rumbaugh, J.: „The Unified Modeling Language Reference Manual“. ADDISON-WESLEY, 1999. ISBN 0-201-30998-X
2. Booch, G., Jacobson, I., Rumbaugh, J.: „The Unified Modeling Language User Guide“. ADDISON-WESLEY, 1999. ISBN 0-201-57168-4
3. Molhanec, Martin: „UML – několik kritických poznámek“, Tvorba software 2002, TANGER, Ostrava 2002
4. Polák, J., Merunka, V., Carda, A.: „Umění systémového návrhu“, GRADA, Praha 2002